A Simple Baseline for Travel Time Estimation using Large-Scale Trip Data

Hongjian Wang[*], Yu-Hsuan Kuo[†], Daniel Kifer[†], Zhenhui Li[*]
[*]College of Information Sciences and Technology, Pennsylvania State University
[†]Department of Computer Science and Engineering, Pennsylvania State University
[*]{hxw186,jessieli}@ist.psu.edu,[†]{yzk5145,dkifer}@cse.psu.edu

ABSTRACT

The increased availability of large-scale trajectory data provides rich information for the study of urban dynamics. For example, New York City Taxi & Limousine Commission regularly releases source/destination information of taxi trips, where 173 million taxi trips released for Year 2013 [1]. Such a big dataset provides us potential new perspectives to address the traditional traffic problems. In this paper, we study the travel time estimation problem. Instead of following the traditional route-based travel time estimation, we propose to simply use a large amount of taxi trips without using the intermediate trajectory points to estimate the travel time between source and destination. Our experiments show very promising results. The proposed big data-driven approach significantly outperforms both state-of-the-art route-based method and online map services. Our study indicates that novel simple approaches could be empowered by the big data and these approaches could serve as new baselines for some traditional computational problems.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data mining; H.4.0 [**Information Systems Applications**]: General

Keywords

Travel time estimation, big data, baseline

1. INTRODUCTION

The positioning technology is widely adopted into our daily life, and a significant amount of trajectory data are collected. While many existing methods try to outdo each other in terms of complexity and algorithmic sophistication, in the spirit of "big data beats algorithms", we study whether some simple baseline methods can be empowered by taking the benefit of big data. We revisit a traditional travel time estimation problem, which estimates the travel time between an origin and a destination.

Existing travel time estimation approaches mostly fall into the line of *route-based* method [2, 3, 4]. Given a source and a destination, these methods first identify a route and then estimate the travel time for this route by aggregating the travel time spent on each segment (or subpath) based on historical trajectories. Such a route-based method faces two major challenges.

SIGSPATIAL'16, October 31-November 03, 2016, Burlingame, CA, USA
© 2016 ACM. ISBN 978-1-4503-4589-7/16/10...$15.00
DOI: http://dx.doi.org/10.1145/2996913.2996943

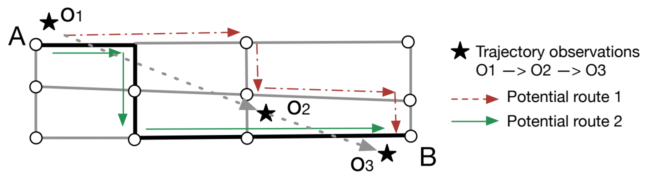

Figure 1: Use historical trajectory $o_1 \rightarrow o_2 \rightarrow o_3$ to estimate the travel time from A to B.

Route mapping. The route-based method needs to obtain the travel time from historical trajectories. But the actual trajectory data may not map well onto the roads due to GPS positioning errors, or imprecise road network data, or the time gap between two adjacent observations. For example, in Figure 1, suppose we are estimating the travel time from A to B. We will first identify a route (the black road segments). Next, to estimate the travel time for this route, we need to know the travel time spent on each segment. And the time spent on each segment is obtained from historical trajectories. One historical trajectory $o_1 \rightarrow o_2 \rightarrow o_3$ is shown on Figure 1. However, it is non-trivial to map these GPS points to the actual road segments. Two potential routes are plotted in Figure 1 and both could be taken by the observed trajectory. In addition, we often have missing data between two consecutive data points. Therefore, it is not clear what is the actual route taken by an observed trajectory.

Data sparsity. Trajectory data are sparse. Even with millions of trip observations, there are a lot of road segments not covered by any trajectory because of the non-uniform spatial distribution of GPS locations in the city. In the Shanghai taxi data used in our experiment, even with over 300 million GPS records, more than 50% of road segments are not covered by any trajectory. The inference of travel time is not accurate due to limited number of trajectories covering the road segment. In addition to the spatial sparseness, the trajectory data are temporally sparse too. Even if a road segment is covered by a few trajectories, these trajectories may not be sufficient to estimate the dynamic travel time that varies under different conditions (e.g., peak time and weekends).

To avoid these challenges in the route-based method, we propose a simple baseline method to estimate the travel time. Suppose we are to predict travel time from A to B and we have hundreds of historical trips from A to B. In this case, we do not need any intermediate points, the average travel time of these historical trips could be a reasonable estimation of the travel time from A to B.

In the real-world setting, even with a large database of millions of trips, it is not easy to get sufficient number of trips with exactly the same origin and destination. Therefore, we look for neighboring trips with a nearby origin and destination to estimate the travel time and we call our method as *neighbor-based method*. However, simply by taking the average of neighboring trips will yield a poor performance because of variance in traffic at different times. For example, the travel time during the peak hour could be much longer compare with that in the midnight; there could also be abnormal traffic changes due to holidays or traffic jams. Therefore, we further propose to consider such traffic dynamics by making use of traffic periodicity and making timely adjustments based on recent traffic patterns.

In summary, the contributions of this paper are as follows. First, we propose to estimate the travel time using neighboring trips from the large-scale historical data. To the best of our knowledge, this is the first work to estimate travel time without computing the routes. Second, we improve our neighbor-based approach by addressing the dynamics of traffic conditions. Finally, our experiments are conducted on two large-scale real datasets. We show that our method can outperform state-of-the-art methods as well as online map services (Bing Maps and Baidu Maps).

The rest of the paper is organized as follows. Section 2 defines the problem and provides an overview of the proposed approach. Sections 3 discusses our method to weight the neighboring trips. We present our experimental results in Section 4 and conclude the paper in Section 5.

2. PROBLEM OVERVIEW

A **trip** $\mathbf{p}_i$ is defined as a 5-tuple $(o_i, d_i, s_i, l_i, t_i)$, which consists of the origin location o_i, the destination location d_i and the starting time s_i. Both origin and destination locations are GPS coordinates. We use l_i and t_i to denote the distance and travel time for this trip, respectively. Note that, here we assume the intermediate locations of the trips are not available. In the real applications, it is quite possible that we can only obtain such limited information about trips due to privacy concerns and tracking costs. For example, the largest public taxi data released by New York City [1] does not contain any intermediate GPS points.

PROBLEM 1 (OD TRAVEL TIME ESTIMATION). *Suppose we have a database of trips, $\mathcal{D} = \{\mathbf{p}_i\}_{i=1}^N$. Given a query $\mathbf{q} = (o_q, d_q, s_q)$, our goal is to estimate the travel time t_q with given origin o_q, destination d_q, and departure time s_q, using the historical trips in $\mathcal{D}$.*

An intuitive solution is finding similar trips as the query trip $\mathbf{q}$ and using the travel time of those similar trips to estimate the travel time for $\mathbf{q}$. The problem can be decomposed to two sub-problems: (1) how to define similar trips; and (2) how to aggregate the travel time of similar trips. Here, we name similar trips as **neighboring trips** (or simply **neighbors**) of query trip $\mathbf{q}$. Note that aggregating is not trivial because of varying starting times and traffic conditions.

We call trip $\mathbf{p}_i$ a neighbor of trip $\mathbf{q}$ if the origin (and destination) of $\mathbf{p}_i$ are spatially close to the origin (and destination) of $\mathbf{q}$. Thus, the set of neighbors of $\mathbf{q}$ is defined as:

$$\mathcal{N}(\mathbf{q}) = \{\mathbf{p}_i \in \mathcal{D} | dist(o_i, o_q) \leq \tau \text{ and } dist(d_i, d_q) \leq \tau\}, \quad (1)$$

where $dist()$ is the Euclidean distance of two given points.

With the definition of neighbors, a baseline approach is to take the weighted average travel time of these trips as the estimation:

$$\widehat{t}_q = \frac{1}{|\mathcal{N}(\mathbf{q})|} \sum_{\mathbf{p}_i \in \mathcal{N}(\mathbf{q})} w_i t_i. \quad (2)$$

For each neighboring trip $\mathbf{p}_i$ of $\mathbf{q}$ we define the *scaling factor* w_i calculated from the speed reference, so that $w_i t_i \approx t_q$. The scaling factor w_i enables us to model the dynamic traffic conditions across different time.

In the following section, we will discuss the technical details of our method.

3. CAPTURING THE TEMPORAL DYNAMICS OF TRAFFIC CONDITIONS

As we discussed earlier, it is not appropriate to simply take the average of all the neighbors of $\mathbf{q}$ because of traffic conditions vary at different times. Now the question is, how can we derive a *temporal scaling reference* to correspondingly adjust travel time on the neighboring trips.

We first define the **scaling factor** of a neighboring trip $\mathbf{p}_i$ on query trip $\mathbf{q}$ as $w_i = \frac{t_q}{t_i}$. One way to estimate s_i is using the speed of $\mathbf{p}_i$ and $\mathbf{q}$. Let v_i and v_q be the speed of trip $\mathbf{p}_i$ and $\mathbf{q}$. Since we pick a small τ to extract neighboring trips of $\mathbf{q}$, it is safe to assume that $\forall \mathbf{p}_i \in \mathcal{N}(q)$, $l_i \simeq l_{\mathbf{q}}$. With this assumption, we have:

$$w_i = \frac{t_q}{t_i} = \frac{l_q/v_q}{l_i/v_i} \approx \frac{v_i}{v_q}.$$

However, v_q is unknown, so we need to estimate $\frac{v_i}{v_q}$. Since the average speed of all trips are stable and readily available, we try to build a bridge from the actual speed ratio to the corresponding average speed ratio for any given two trips. One solution is to assume the ratio between v_q and v_i approximately equals to the ratio between the average speed of all trips at s_q and s_i. Formally, let $V(s)$ denote the average speed of all trips at timestamp s, we have an approximation of w_i as

$$w_i \approx \frac{v_i}{v_q} \approx \frac{V(s_i)}{V(s_q)}. \quad (3)$$

Considering such scaling factors using temporal speeds as the reference, we can estimate the travel time of $\mathbf{q}$ using the neighboring trips as follows:

$$\widehat{t}_q = \frac{1}{|\mathcal{N}(\mathbf{q})|} \sum_{\mathbf{p}_i \in \mathcal{N}(\mathbf{q})} t_i \cdot \frac{V(s_i)}{V(s_q)}. \quad (4)$$

We show the effectiveness of this predictor in Figure 2. Each point in the figure is a target trip. The prediction is plotted against the actual trip travel time. We can see that the prediction is close to the actual value. This indirectly implies the validity of assumptions we made previously.

In order to compute the average speed $V(s_i)$, we need to collect all the trips in $\mathcal{D}$ which started at time s_i. However, for the query trip $\mathbf{q}$, the starting time s_q may be the current time or some time in the future. Therefore, no trips in $\mathcal{D}$ have the same starting time as $\mathbf{q}$. In the following, we discuss two approaches to predict $V(s_q)$ using the available data.

3.1 Relative Temporal Speed Reference

In this section, we assume that $V(s)$ exhibits a regular daily or weekly pattern. We fold the time into a relative time window $\mathcal{T}_{rela} = \{1, 2, \cdots, T\}$, where T is the assumed periodicity. For example, using a weekly pattern with 1 hour as the basic unit, we

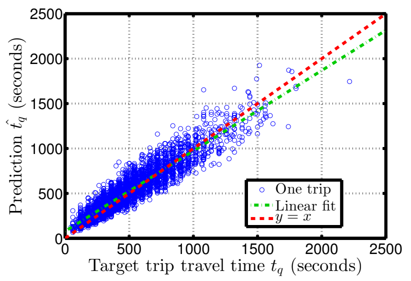

Figure 2: Estimated travel time against actual travel time. Each point is a trip, where the estimation $\hat{t_q}$ is plotted against t_q. The green line is the linear fitting of the points, which is closer to the red line $y = x$. This means the prediction is close to the actual value.

have $T = 7 \times 24 = 168$. Using this relative time window, we represent the average speed of the k-th time slot as $V_k, \forall k \in \mathcal{T}_{rela}$. We call $\{V_k | k \in \mathcal{T}_{rela}\}$ *relative temporal reference*.

We use k_i to denote the time slot to which s_i belongs. As a result, we can write $V(s_i) = V_{k_i}$. To compute V_{k_i}, we collect all the trips in $\mathcal{D}$ which fall into the same time slot as $\mathbf{p}_i$ and denote the set as $S(\mathbf{p}_i)$. Then, we have

$$V_{k_i} = \frac{1}{|S(\mathbf{p}_i)|} \sum_{\mathbf{p}_j \in S(\mathbf{p}_i)} \frac{l_j}{t_j}. \tag{5}$$

The relative speed reference mainly has the following two advantages. First, the relative speed reference is able to alleviate the data sparsity issue. By folding the data into a relative window, we will have more trips to estimate an average speed with a higher confidence. Second, the computation overhead of relative speed reference is small, and we could do it offline.

3.2 Absolute Temporal Speed Reference

In the previous section, the relative speed reference assumes the speed follows daily or weekly regularity. However, in real scenario, there are always irregularities in the traffic condition. For example, during national holidays the traffic condition will significantly deviate from the usual days. Therefore, assuming we have enough data, it would be more accurate if we could directly infer the average speed at any time slot t from the historical data. Following this idea, we propose to directly capture the traffic condition at different time slots with an *absolute temporal speed reference*. To this end, we partition the original timeline into time slots based on a certain time interval (i.e., 1 hour). All historical trips are mapped to the absolute time slots $\mathcal{T}_{abs} = \{1, 2, 3, \cdots\}$ accordingly, and the average speed $\{V_k | k \in \mathcal{T}_{abs}\}$ are calculated as the absolute temporal speed reference.

The challenge in absolute temporal speed reference is that for a given query trip with starting time s_q in the near future (e.g. next hour), we need to estimate the speed reference $V(s_q)$. We estimate $V(s_q)$ with seasonal ARIMA model by considering factors such as the average speed of previous hours, seasonality, and random noise. Formally, given the time series of average speed: $\{V_1, V_2, \ldots, V_M\}$, our goal is to compute V_{M+1} as follows:

$$V_{M+1} = f(V_1, \ldots, V_M). \tag{6}$$

In our problem, the average speed V_t exhibits a strong weekly pattern. Thus, instead of directly applying the ARIMA model to $\{V_t\}$, we first compute the sequence of seasonal difference $\{Y_t\}$:

$$Y_t = V_t - V_{t-T}, \tag{7}$$

where T is the period (e.g., one week). Note that our ARIMA model with the seasonal difference is a special case of the more general class of Seasonal ARIMA (SARIMA) model for time series analysis. We refer interested readers to [5] for detailed discussion about the model.

Then, we apply the ARIMA model to $\{Y_t\}$:

$$\left(1 - \sum_{i=1}^{p} \phi_i L^i\right) (1 - L)^d Y_t = \left(1 + \sum_{i=1}^{q} \theta_i L^i\right) \epsilon_t, \tag{8}$$

where L is the *lag operator*.

Since the last term ϵ_t is white noise, whose value is unknown but the expectation $E(\epsilon_t) = 0$, we have estimator $\hat{Y}_t = Y_t - \epsilon_t$. Together with Equation (7), we have

$$\hat{V}_t = \hat{Y}_t + V_{t-T}. \tag{9}$$

4. EXPERIMENT

4.1 Evaluation Settings

We conduct experiments on datasets from two different countries to evaluate our approach.

NYC Taxi dataset [1] contains $173,179,771$ taxi trips from 2013/01/01 to 2013/12/31. Each trip contains information about pickup location and time, drop off location and time, trip distance, fare amount, etc. We use the subset of trips within the borough of Manhattan (the boundary is obtained from wikimapia.org), which has $\mathbf{127,534,711}$ trips. On average, we have $349,410$ trips per day.

Shanghai Taxi dataset [6] contains the trajectories of $2,600$ taxis during two months in 2006, which enables us to compare with the existing route-based methods. In total we have over 300 million GPS records. A GPS record has following fields: vehicle ID, speed, longitude, latitude, occupancy, and timestamp. To retrieve taxi trips in this dataset, we rely on the occupancy bit of GPS records. This occupancy bit is 1 if there are passengers on board, and 0 otherwise. For each taxi, we define a trip as consecutive GPS records with occupancy equal to 1. We get $5,815,470$ trips after processing the raw data.

Methods for evaluation. We systematically compare the following methods: linear regression (`LR`), neighbor average (`AVG`), temporally weighted neighbors (`TEMP`), segment-based estimator (`SEGMENT`), subpath-based estimator (`SUBPATH`), online map service (`BING` and `BAIDU`). The `TEMP` is our proposed method. More specifically, we name the method using relative-time speed reference as $\texttt{TEMP}_{\texttt{rel}}$ (Section 3.1), and the method using absolute temporal reference as $\texttt{TEMP}_{\texttt{abs}}$ (Section 3.2). The `SEGMENT` is a baseline method used in [7], which has a drawback that the transition time at intersections cannot be captured. Therefore, Wang et al. [7] propose the `SUBPATH` to estimate the target route with concatenated subpaths, where each subpath is defined as multiple road segments. We use Bing Maps for NYC taxi dataset and Baidu Maps for Shanghai dataset.

Evaluation metrics. We use mean absolute error (MAE), mean relative error (MRE), median absolute error (MedAE) and median relative error (MedRE) to evaluate the travel time estimation methods:

$$\text{MAE} = \frac{\sum_i |y_i - \hat{y}_i|}{n}, \text{MRE} = \frac{\sum_i |y_i - \hat{y}_i|}{\sum_i y_i},$$

$$\text{MedAE} = median(|y_i - \hat{y}_i|), \text{MedRE} = median\left(\frac{|y_i - \hat{y}_i|}{y_i}\right),$$

where $\hat{y}_i$ is the travel time estimate, y_i is the ground truth, and *median* returns the median value of a vector. We add the median error as evaluation metric as well, because there are anomalous trips in the dataset that differ significantly from most trips.

4.2 Performance on NYC Data

We use trips from January to November as training and the ones in December as testing. As for the `BING` method, due to the limited quota, we sample $260k$ trips in December as testing. The overall accuracy comparison are shown in Table 1. Since NYC data only have endpoints of trips, we cannot compare our method with route-based methods. We will compare with theses methods using Shanghai data in Section 4.3.

Table 1: Overall Performance on NYC Data

Method	MAE (s)	MRE	MedAE (s)	MedRE
`LR`	194.604	0.2949	164.820	0.3017
`AVG`	178.459	0.2704	120.834	0.2345
$\texttt{TEMP}_{\texttt{rel}}$	149.815	0.2270	**97.365**	**0.1890**
$\texttt{TEMP}_{\texttt{abs}}$	**143.311**	**0.2171**	98.780	0.1907
Comparison with Bing Maps (260K testing trips)				
`BING`	202.684	0.3157	134.000	0.2718
`BING(traf)`	242.402	0.3776	182.000	0.3395
$\texttt{TEMP}_{\texttt{abs}}$	**135.365**	**0.2108**	**94.940**	**0.1839**

We first observe that our method is better than the linear regression (`LR`) baseline. This is expected because `LR` is a simple baseline which does not consider the origin and destination locations.

Considering temporal variations of traffic condition improves the estimation performance. All the `TEMP` methods have significantly lower error compared with the `AVG` method. The improvement of $\texttt{TEMP}_{\texttt{abs}}$ over `AVG` is about 35 seconds in MAE, i.e. 20% improvement. Furthermore, the absolute speed reference ($\texttt{TEMP}_{\texttt{abs}}$) is better than the relative (i.e., weekly) speed reference ($\texttt{TEMP}_{\texttt{rel}}$), because the traffic condition does not strictly follow the weekly pattern, but has some irregular days such as holidays.

Comparing with `BING` on the $260k$ testing trips, $\texttt{TEMP}_{\texttt{abs}}$ significantly outperforms `BING` by 67 seconds. `BING` underestimates the trips without considering traffic, where 64.53% testing trips are underestimated. However, `BING(traf)` by considering traffic (the query was sent to the API at the same time and the same day of the week), 75.02% testing trips are overestimated.

4.3 Performance on Shanghai Data

In this section, we conduct experiments on Shanghai taxi data to compare our method with `SEGMENT` and `SUBPATH` methods. Both `SEGMENT` and `SUBPATH` methods use the travel time on individual road segments or subpaths to estimate the travel time for a query trip. Due to data sparsity, we cannot obtain travel time for every road segments. In our experiment, to avoid the missing value issue, we only select the testing trips that have values on every segments of the trip. The testing set contains $2,138$ trips in total. We use the same training set and testing set for different methods.

The comparison among different methods is shown in Table 2. Our neighbor-based method significantly outperform other methods. The simple method `AVG` is 17 seconds better than `BAIDU` in terms of MAE. By considering temporal factors, method $\texttt{TEMP}_{\texttt{rel}}$ and $\texttt{TEMP}_{\texttt{abs}}$ further outperform `AVG` method. `SUBPATH` method outperforms `SEGMENT` method, which is consistent with previous work [7]. However, `SUBPATH` is still 21 seconds worse than our method $\texttt{TEMP}_{\texttt{rel}}$ method in terms of MAE. Our method can be considered as a special case `SUBPATH` method, where we use the whole paths from the training data to estimate the travel time for a testing trip. Given enough number of whole paths, it is better to use the whole paths instead of subpaths.

Table 2: Overall Performance on Shanghai Data

Method	MAE (s)	MRE	MedAE (s)	MedRE
`LR`	130.710	0.6399	138.796	0.6173
`BAIDU`	111.484	0.5451	73.001	0.5001
`SEGMENT`	119.833	0.5866	84.704	0.4947
`SUBPATH`	113.566	0.5560	75.913	0.4820
`AVG`	94.202	0.4615	60.183	0.3739
$\texttt{TEMP}_{\texttt{rel}}$	**92.428**	**0.4527**	**55.317**	**0.3678**

Since the Shanghai taxi data are much more sparse than NYC data, we do not show the results of $\texttt{TEMP}_{\texttt{abs}}$ on Shanghai dataset. The main idea here is to show that $\texttt{TEMP}_{\texttt{rel}}$ can outperform the existing methods, and with more data our other approaches should outperform the $\texttt{TEMP}_{\texttt{rel}}$ as well as shown in NYC data.

5. CONCLUSION

This paper demonstrates that one can use large-scale trip data to estimate the travel time between an origin and a destination in a very efficient and effective way. Our proposed method retrieves all the neighboring historical trips with the similar origin and estimation locations and estimate the travel time using those neighboring trips. We conduct experiments on two large-scale real-world datasets and show that our method can greatly outperform the state-of-the-art methods and online map services.

Acknowledgements

The work was supported in part by NSF award #1544455, #1618448, and #1054389. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies.

6. REFERENCES

[1] C. Whong, "Foiling nyc's taxi trip data: http://chriswhong.com/open-data/foil_nyc_taxi/."

[2] E. Kanoulas, Y. Du, T. Xia, and D. Zhang, "Finding fastest paths on a road network with speed patterns," in *IEEE International Conference on Data Engineering*, 2006.

[3] H. Gonzalez, J. Han, X. Li, M. Myslinska, and J. P. Sondag, "Adaptive fastest path computation on a road network: a traffic mining approach," in *International Conference on Very large Data Bases*, 2007, pp. 794–805.

[4] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang, "T-drive: driving directions based on taxi trajectories," in *ACM SIGSPATIAL*, 2010, pp. 99–108.

[5] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2014.

[6] H. Wang, Y. Zhu, and Q. Zhang, "Compressive sensing based monitoring with vehicular networks," in *INFOCOM, 2013 Proceedings IEEE*. IEEE, 2013, pp. 2823–2831.

[7] Y. Wang, Y. Zheng, and Y. Xue, "Travel time estimation of a path using sparse trajectories," in *ACM SIGKDD*, 2014.